[Smart and Scalable Urban Traffic Control] http://www.wiomax.com/traffic

Schedule-Driven Coordination for Real-Time Traffic Network Control

Xiao-Feng Xie, Stephen F. Smith and **Gregory J. Barlow**
The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
{xfxie, sfs, gjb}@alumni.cmu.edu

Abstract

Real-time optimization of the dynamic flow of vehicle traffic through a network of signalized intersections is an important practical problem. In this paper, we take a decentralized, schedule-driven coordination approach to address the challenge of achieving scalable network-wide optimization. To be locally effective, each intersection is controlled independently by an on-line scheduling agent. At each decision point, an agent constructs a schedule that optimizes movement of the observable traffic through the intersection, and uses this schedule to determine the best control action to take over the current look-ahead horizon. Decentralized coordination mechanisms, limited to interaction among direct neighbors to ensure scalability, are then layered on top of these asynchronously operating scheduling agents to promote overall performance. As a basic protocol, each agent queries for newly planned output flows from its upstream neighbors to obtain an optimistic projection of future demand. This projection may incorporate non-local influence from indirect neighbors depending on horizon length. Two additional mechanisms are then introduced to dampen "nervousness" and dynamic instability in the network, by adjusting locally determined schedules to better align with those of neighbors. We present simulation results on two traffic networks of tightly-coupled intersections that demonstrate the ability of our approach to establish traffic flows with lower average vehicle wait times than both a simple isolated control strategy and other contemporary coordinated control strategies that use moving average forecast or traditional offset calculation.

Introduction

Traffic congestion is a practical problem resulting in substantial delays and extra fuel costs for drivers. It is generally recognized that improvements to traffic signal control provide the biggest payoff for reducing congestion on surface streets, and that adaptive control strategies capable of responding to traffic conditions in real-time hold the most potential for improvement. However, how to achieve scalable network-wide optimization remains a challenging problem.

For a single intersection, the basic challenge is to utilize high-resolution (e.g., second-by-second) flow data obtained from on-line surveillance techniques (Sharma, Bullock, and Bonneson 2007) and to take advantage of rapid revisions (e.g., every 0.5 seconds) to the intersection's executing signal plan (Cai, Wong, and Heydecker 2009). Offline optimization techniques, e.g., SYNCHRO (Husch and Albeck 2006), in contrast, generate signal plans that reflect low-resolution average flows. Reinforcement learning methods have been applied to try to find policies for mapping local observations in a *prediction horizon* to signal actions for intersections (Richter, Aberdeen, and Yu 2006; Cai, Wong, and Heydecker 2009). However, these methods are often slow to converge and difficult to apply in real time if traffic flows are changing frequently. Other work has focused on online planning algorithms, e.g., COP (Sen and Head 1997), ALLONE-S (Porche and Lafortune 1999), and OPAC (Gartner, Pooran, and Andrews 2002), which proceed according to a rolling horizon and attempt to find a good signal sequence in a *planning horizon* for the current observation at each decision point (Ross et al. 2008). Unfortunately, most existing algorithms for the traffic control problem are not real-time feasible for realistic planning horizons (Papageorgiou et al. 2003) and are often forced to plan at coarse time resolutions (e.g., 5 seconds), due to the inefficiency of searching in an exponential planning search space.

At the network level, the major challenge is that some intersections are tightly-coupled, even if they are indirectly connected, since travel times between them can be short and the demands can be high. Congestion that starts from these intersections can spread to neighboring intersections and eventually lead to "gridlock" (Cervero 1986) in the overall network. To address this challenge, coordination mechanisms for managing non-local influence among intersections have also received considerable attention. Centralized (or hierchachical) coordination mechanisms have been proposed for dynamically adjusting offsets (Heung, Ho, and Fung 2005; Gettman et al. 2007) or for imposing constraints on local controllers to match changing traffic patterns, as in RHODES (Mirchandani and Head 2001). However, these mechanisms are inherently susceptible to scalability issues. For example, the network offset adjustment in ACS-Lite (Gettman et al. 2007) has been found to be intractable in real time for only 12 intersections, and existing decentralized constrained optimization methods have issues related to large message sizes and long running times (Junges and Bazzan 2008).

Other approaches have adopted a more decentralized view

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

of coordination, emphasizing mechanisms that use or exchange information to extend the myopic views of individual intersection control methods as they conduct best-response behavior (Jonsson and Rovatsos 2011). One basic way has been to use a forecast of demand beyond the local prediction horizon (Blackburn, Kropp, and Millen 1986). In OPAC, a moving average of historical arrivals from neighbors is used to extend local prediction on each entry road. Communication with more accurate information (Wu, Zilberstein, and Chen 2009) can help for multiagent online planning. In COP, input flows of direct upstream neighbors are used directly for predicting future arrivals to provide greater sensitivity to actual traffic conditions. These predictions, however, do not include the non-local influences of indirect neighbors.

Starting from this decentralized view of traffic network control and following the perspective of recent work in multi-agent coordination (Lesser, Decker, and Wagner 2004; Smith et al. 2007), this paper proposes a decentralized, schedule-driven coordination approach to the real-time traffic network control problem. At the core of our approach is a recently developed SCHedule-driven approach to Intersection Control (SchIC) (Xie et al. 2011). Equipped with compatible interfaces to the inputs and outputs assumed by traditional online planning approaches to intersection control, SchIC defines an abstract *scheduling search space* that enables efficient computation of near-optimal solutions in polynomial time, and it has been shown to outperform other state-of-the-art online planning approaches (e.g., COP). To define a basic intersection scheduling agent, SchIC is first generalized for operation in a traffic network, and then combined with network level protocols for exchanging and using non-local schedule information. To ensure scalability, interaction is limited to direct neighbors. In operation, each scheduling agent operates asynchronously and at each decision point requests a projection of scheduled output flows from its direct upstream neighbors. Since this projection essentially increases an agent's look-ahead horizon, it is capable of incorporating non-local impacts from indirect neighbors. This basic protocol is designed to yield coordinated behavior in "perfect" situations where no agents make significant changes to previously computed schedules. Additional mechanisms are then introduced to dampen "nervousness"(Blackburn, Kropp, and Millen 1986) and dynamic instability (Daganzo 1998; Kumar and Meyn 1995) in the traffic network in less than perfect situations. When applied, these coordination mechanisms adjust an agent's local schedule to achieve a better network-level effect.

Problem Definition

We focus on a road network with a traffic light at each intersection. For each intersection with a set of entry and exit roads, the traffic light cycles through a fixed sequence of phases I, and each phase $i \in I$ governs the right of way for a set of compatible movements from entry to exit roads. The next phase of i is $next(i) = (i + 1) \bmod |I|$. For traffic signal control, a *signal sequence* (SS) contains a sequence of phases and associated durations. For the switching process, there are some timing constraints for safety and fairness: the yellow light after each phase i runs for a fixed duration (Y_i), while each phase i has a variable duration (g_i) that can range between a minimum (G_i^{min}) and maximum (G_i^{max}).

We assume that each agent holds a private signal sequence SS_{TL} that controls an intersection for a finite future time, which is periodically updated according to a *rolling horizon*: When the time reaches the end of SS_{TL} (at the next *decision point*), the agent computes a new sequence SS_{ext} to extend SS_{TL}. The objective of the agents is to minimize the total cumulative delay of vehicles traveling through the road network over a time period.

Basic traffic models are also assumed. On each road, spatial distances are transformed into temporal values by dividing the average *free-flow speed*, and a queue of vehicles is discharged in a green phase at the *saturation flow rate* (sfr) after the *start-up lost time* (slt) (Sharma, Bullock, and Bonneson 2007). As in RHODES, we assume *turning proportions* at each intersection are available. In practice, quite accurate parameters can be estimated by using measured data.

We assume that all temporal values have been rounded into numbers of time steps of a discrete time resolution (Δ).

Schedule-Driven Intersection Control (SchIC)

At each decision time, the online planning problem faced by a given intersection agent is to produce a signal sequence SS_{ext} for the next period. This is accomplished by first generating a *control flow* in a scheduling search space that minimizes local cumulative delay given the current observation o of incoming traffic flows, and then applying the timing constraints to obtain feasible SS_{ext}. We first define inputs and outputs, and then describe the core scheduling procedure.

Road Flows, Inflows and Control Flow

Sensed vehicles in a given traffic flow are characterized as a *cluster sequence* $C = (c_1, \cdots, c_{|C|})$, where $|C|$ is the number of clusters in C. Each *cluster* c is defined as $(|c|, arr, dep)$, where $|c|$ is the number of vehicles in c, and arr (dep) gives the expected *arrival* (*departure*) time at the intersection respectively for the first (last) vehicle in c. The clusters in C are ordered by increasing arr values.

The *road flows* entering an intersection then consist of a set of cluster sequences RF, where each $C_{RF,m}$ contains the vehicles traveling toward the intersection on entry road m. In practice, each $C_{RF,m}$ can be obtained using detectors on both sides of m (Sharma, Bullock, and Bonneson 2007), where the travel time on m defines a finite *horizon* (H_m), and the prediction horizon H is the maximum over all roads.

Since it is possible for more than one entry road to have the right of way in a given phase (e.g., a two-way street), the actual traffic flows of interest in determining a signal sequence are the *inflows* IF, which contain cluster sequences that combine traffic flows that can proceed concurrently through the intersection. Formally, $IF = (C_{IF,1}, \cdots, C_{IF,|I|})$, where $C_{IF,i}$ contains those vehicles with the right of way during phase i. These intersection movement patterns are generally known and assumed available by existing control methods (Sen and Head 1997). Given turning proportions for these movement patterns, IF can be obtained through a road-to-phase mapping, i.e., $IF =$

$RtoP(RF)$, where flows on roads are extracted according to turning proportions and assembled into phased-based flows. The prediction horizon H remains the same after the transformation. If each road is only serviced in one phase, as in many real-world intersections, $RtoP$ is trivial.

To reduce the scheduling search space faced by the intersection scheduler, two aggregation techniques that exploit the non-uniform nature of traffic flows are applied on each cluster sequence in IF to form the final observation o. First, arriving vehicles that are within a threshold time gap ($th_g \geq 0$) are merged into a single arriving cluster. Second, vehicles that are expected to join a given queue before it is able to clear the intersection are grouped into an *anticipated queue* cluster (Lämmer and Helbing 2008).

The *control flow* CF for an intersection contains the results of applying a signal sequence that clears all clusters in an observation o. Formally, CF can be represented as a tuple (S, C_{CF}), where S is represented as a sequence of phase indices, i.e., $(s_1, \cdots, s_{|S|})$, C_{CF} contains a corresponding sequence of clusters $(c_{CF,1}, \cdots, c_{CF,|S|})$ that are reorganized from IF. For each k, all vehicles in $c_{CF,k}$ belong to C_{IF,s_k}.

As $RtoP$ is applied, a *road-ratio* function $rr(c, m)$ is used to store the ratio of vehicles from the entry road m, for each cluster c in IF, and hence for each cluster in CF as well. This interface provides the necessary information for coordinating with upstream/downstream neighbors.

Scheduling Strategy

We use a schedule-driven intersection control (SchIC) strategy, by viewing each intersection as a single machine, and each cluster in IF as a non-divisible job. The jobs in $C_{IF,i}$ can only leave the intersection in phase i, and the jth cluster can only leave after the $(j-1)$th one has left. Since the cumulative dalay is only associated with the jobs in IF, a scheduling search space with an efficient state elimination criterion can be formed from the current observation.

The current observation o is defined to contain the current decision time cdt, the phase index cpi and duration cpd of the currently active traffic light phase, and the inflows IF computed for the current prediction horizon H.

In a control flow $CF = (S, C_{CF})$, S can be seen as a *schedule* with $|S| = \sum_{i=1}^{|I|} |C_{IF,i}|$ elements. For a partial schedule S_k, i.e., the first k elements of S, its *schedule status* is defined as X=$(x_1, \cdots, x_{|I|})$, where $x_i \in [0, |C_{IF,i}|]$ counts the number of clusters that have been serviced for phase i. The kth job in C_{CF} comes from the x_{s_k}th cluster in C_{IF,s_k} (although its actual start time might be shifted according to a greedy realization in Algorithm 1, see below).

For each S_k, the corresponding state variables are defined as a tuple, $(X, s, pd, t, d)_k$, where s and pd are the index and duration of the last phase, t is the finish time of the kth job, and d is the cumulative delay for all k jobs.

The state variables of S_k can be updated from those of S_{k-1}, where s_k is known, X_k = (X_{k-1} with $x_{s_k} = x_{s_k} + 1$), and $(pd,\ t,\ d)_k$ are calculated by Algorithm 1 using (s, pd, t, $d)_{k-1}$ and s_k. Algorithm 1 is based on a greedy realization of a planned signal sequence, where $MinSwitch(s, i)$ in Line 3 returns the minimum time required for switching

Algorithm 1 Calculate (pd, t, d) of S_k (and obtains $c_{CF,k}$)

Require: 1) (s, pd, t, d) of S_{k-1}; 2) s_k
1: $i = s_k$; c = (the x_ith job in $C_{IF,i}$)
2: **if** $(s \neq i)$ and $(pd < G_s^{min})$ **then** $t = t + (G_s^{min} - pd)$
3: $pst = t + MinSwitch(s, i)$ {Permitted start time of c}
4: $ast = \max(arr(c), pst)$ {Actual start time of c}
5: **if** $(s \neq i)$ and $(pst > arr(c))$ **then** $ast = ast + slt_i$
6: $t = ast + dep(c) - arr(c)$ {Actual finish time of c}
7: **if** $(s \neq i)$ or $(arr(c) - pst > SwitchBack(s))$
8: **then** $pd = t - pst$ **else** $pd = pd + (t - pst)$
9: $d = d + |c| \cdot (ast - arr(c))$ {Total cumulative delay}
10: **return** (pd, t, d) of S_k {$c_{CF,k} = (|c|, ast, t)$}

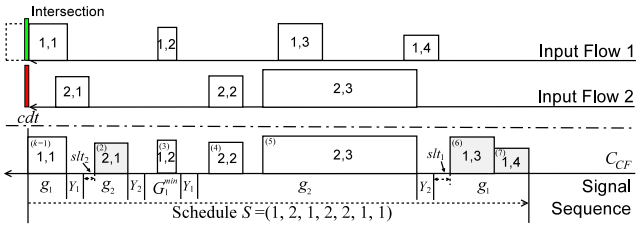

Figure 1: A schedule of jobs (clusters), the corresponding C_{CF}, and a planned signal sequence

from the phase index s to i, Line 2 ensures each phase s is not shorter than the minimum G_s^{min}, slt_i in Line 5 is the *start-up lost time* for clearing the queue in the phase i, and $SwitchBack(i)$ in Line 7 is the minimum time required for that the traffic light returns to the phase index i. Both $MinSwitch$ and $SwitchBack$ only include yellow and minimum green times during the switching process. Compared to the corresponding cluster c in IF, the delay time of the kth job in C_{CF} is $(ast_k - arr(c))$ (Line 9).

Figure 1 shows a schedule, a planned signal sequence, and C_{CF}. Note that for a given observation, a control flow might encapsulate different signal sequences due to the presence of slack time in the schedule. For example, the phase that services the cluster (2,1) might be prolonged a little without changing the cumulative delay of the control flow. This slack time can be useful for coping with uncertainty in traffic flow.

Based on the observation o, a forward recursion, dynamic programming process is used to obtain a near optimal solution S^* among possible schedules that minimizes the total cumulative delay. To retain efficiency, the states are grouped by using (X, s), only one state with the minimal d value is stored and other states in the group (or other branches in the context of a decision tree) are eliminated. The remaining state variables of the state are stored as a *value row* (pd, t, d, s_o), where an additional value s_o is used for tracking back to the previous s (as will be used in Algorithm 4).

Algorithm 2 recursively calculates the value rows in all required state groups (X, s). Two unique X arrays, i.e., X_{empty} and X_{full}, which have $x_i = 0$ and $x_i = |C_{IF,i}|$ for $\forall i$, correspond to the empty and full status, respectively. Initially, only the state group (X_{empty}, cpi) has the value row $(cpd, cdt, 0, -)$. For all other state groups (X, s), their value

Algorithm 2 Forward recursion process

1: (pd, t, d, s_O) of (X_{empty}, cpi)= $(cpd, cdt, 0, -)$
2: **for** $k = 1$ to $|S|$ **do**
3: Collect the set $\underline{X}_k = \{X : \sum_{i=1}^{|I|} x_i \equiv k\}$
4: **for** $\forall X \in \underline{X}_k, \forall s \in [1, |I|]$ **do**
5: **if** $x_s > 0$ **then** Execute Algorithm 3 for (X, s)
6: **end for**
7: **end for**
8: **return** The solution S^* by using Algorithm 4

Algorithm 3 Calculate (pd, t, d, s_O) of (X, s)

1: X_O = (X with $x_s = x_s - 1$), $d_{min} = \infty$
2: **for** $s_O = 1$ to $|I|$ **do**
3: **if** (X_O, s_O) exists **then**
4: $(pd_O, t_O, d_O) = (pd, t, d)$ of (X_O, s_O)
5: (pd, t, d) = Alg. 1, given (s_O, pd_O, t_O, d_O) and s
6: **if** $(d < d_{min})$ **then**
7: Store (pd, t, d, s_O) for (X, s); $d_{min} = d$
8: **end if**
9: **end if**
10: **end for**

rows are then calculated in Algorithm 2 and stored. Using the set $\underline{X}_k$ in Line 3 is a naive way of ensuring that all input state groups are available for Algorithm 3, which adds the kth element s to possible S_{k-1}. The condition $x_s > 0$ in Line 5 is used for ensuring the kth job is available.

Algorithm 2 has at most $|I|^2 \cdot \prod_{i=1}^{|I|}(|C_{IF,i}| + 1)$ state updates, where $|C_{IF,i}| \leq H$, and each state update in Lines 4-8 of Algorithm 3 can be executed in constant time. It is polynomial in H since $|I|$ is limited for each intersection in real world. Note that $|C_{IF,i}|$ is normally much smaller than H. The *planning horizon* is implicitly available as the maximum finish time of all schedules, which might be much larger than H in congested traffic conditions.

The solution S^* is tracked back using Algorithm 4. The corresponding C^*_{CF} and $PD^* = (pd_1, \cdots, pd_{|S|})$ are obtained from Algorithm 1. The tuple (S^*, C^*_{CF}, PD^*) is stored until it is replaced in the next scheduling iteration.

Commitment Process

The role of the commitment process is to determine what initial portion of the just computed schedule (SS_{ext}) to append to the signal sequence (SS_{TL}) that is controlling the intersection. There is a basic trade-off for deciding the duration of SS_{ext}. A shorter duration enables quicker response to changes in flow information, whereas a longer duration leads to a more stable control flow for downstream agents.

For simplicity, we only consider whether to extend the current phase or move to the next phase. An extension proposal is first made by using the first job in C^*_{CF}, called c_1, if available. There are two extension choices: 1) $ext = 0$, if $|S^*| \equiv 0$, or $s^*_1 \neq cpi$, or $arr(c_1) \geq SwitchBack(cpi)$; otherwise 2) $ext = \min(dep(c_1) - cdt, th_{ext})$, where th_{ext} is the upper limit to favor a quick responsive capability.

Note that SS_{TL} must satisfy all timing constraints. This requirement is ensured by applying a *repair* rule (Porche and Lafortune 1999): If $ext > 0$, the current phase is extended to $cpd = \min(cpd + ext, G^{max}_{cpi})$. If $cpd \equiv G^{max}_{cpi}$ or $ext \equiv 0$, the current phase is terminated, and the next phase is added for a minimum green time after the yellow time.

Algorithm 4 Retrieve the solution S^*

1: $X = X_{full}$; $s = \arg\min_s (d \text{ of } (X, s))$
2: **for** $k = |S|$ to 1 **do**
3: $s_k = s$; $s = s_O$ of (X, s); $x_{s_k} = x_{s_k} - 1$
4: **end for**
5: **return** $S^* = (s_1, \cdots, s_{|S|})$

Neighbor Coordination Mechanisms

The local observations of an isolated agent only consider vehicles that have arrived in the detection zones of the intersection's entry roads. If the entry roads are short, the prediction horizon will be short and the agent is susceptible to myopic decisions that look good locally but not globally. To counteract this possibility we augment the above intersection control strategy with explicit coordination mechanisms.

Specifically, we focus on introducing decentralized coordination mechanisms between direct neighbors. The low overhead of this approach allows for coordination in real-time. Following the insights of existing coordination frameworks, each agent remains highly autonomous. In our case, the intersection control strategy always runs at the base level to tailor local action to local information. Although this setting certainly restricts possible choices, simple coordination mechanisms can still be introduced to improve the overall performance significantly, as we will show below.

Our approach includes a basic protocol and two additional coordination mechanisms. The basic protocol, similar to a social law, is defined in the sense that the basic behavior of agents will be coordinated in perfect situations. Additional coordination mechanisms are then applied to handle two nontrivial mis-coordinated situations in the network: "nervousness" and dynamic instability.

Basic Operations

Some operations are used for simplifying the description.

The operation $C \cap [t_1, t_2]$ forms a new cluster sequence that only contains (partial) clusters belonging to $[t_1, t_2]$, where a cluster is cut if it spans the boundary. If a cluster c is *cut* into two parts, the number of vehicles in c is divided according to the proportions of their respective durations.

The operation $(S, C) \cap [t_1, t_2]$ forms (S', C'), where $C' = C \cap [t_1, t_2]$, S' is a subsequence of S, where each element is removed if the corresponding cluster in C is totally removed.

The *Unschedule*(t_1, t_2) operation removes the clusters in $[t_1, t_2]$ from (S^*, C^*_{CF}), and releases all corresponding (partial) clusters that are not in C^*_{CF} to form a new IF.

The *Shift*(C, t) operation shifts the arr and dep values of all clusters in the sequence C forward in time by t.

Algorithm 5 Obtain an optimistic non-local observation

1: **for** Each entry road m **do**
2: $UpAgent = UpstreamAgent(m)$
3: Request C_{OF} from $UpAgent$ using (cdt, m, H_{ext})
4: $Shift(C_{OF}$, the free travel time on the road $m)$
5: Append C_{OF} into $C_{RF,m}$
6: **end for**
7: $IF = RtoP(RF)$ {road-to-phase mapping}

Algorithm 6 Return C_{OF} for a message (cdt, n, H_{ext})

1: $(S_{OF}, C_{OF}) = (S^*, C^*_{CF}) \cap [cdt, cdt + H_{ext}]$
2: **for** $k = 1$ to $|C_{OF}|$ **do**
3: $TurnRatio = \sum_m (rr(c_{OF,k}, m) \cdot tp(s_{OF,k}, m, n))$
4: $|c_{OF,k}| = |c_{OF,k}| \cdot TurnRatio$
5: **end for**

Algorithm 7 Obtain a fully feasible control flow (S^*, C^*_{CF})

1: $S^* = C^*_{CF} = \varnothing$
2: **repeat**
3: $t_{vio} = \infty$; (S, C_{CF}, PD) = SchIC(o)
4: Append (S, C_{CF}) into (S^*, C^*_{CF})
5: **for** $k = 1$ to $|C_{CF}|$ **do**
6: **if** $pd_k > G^{max}_{s_k}$ **then**
7: $t_{vio} = dep(c_{CF,k}) - (G^{max}_{s_k} - pd_k)$
8: **if** $t_{vio} < arr(c_{CF,k})$ **then** $t_{vio} = dep(c_{CF,k-1})$
9: $Unschedule(t_{vio}, \infty)$ {also update IF}
10: $t_c = t_{vio} + Y_{s_k}$; $cpi = next(s_k)$; $cpd = 0$
11: **break** {break the for-loop}
12: **end if**
13: **end for**
14: **until** $< t_{vio} \equiv \infty >$

Optimistic Non-Local Observation

For each agent, the basic protocol with its upstream agents is achieved by using an *optimistic non-local observation*, as shown in Algorithm 5. For each entry road m, the corresponding upstream agent $UpAgent$ is obtained. The agent then sends each $UpAgent$ a request message (cdt, m, H_{ext}), where H_{ext} is the maximum *horizon extension*, in order to obtain a planned *outflow* C_{OF} from $UpAgent$. Upon receipt of C_{OF}, the downstream agent adds an offset time — the average travel time between the two agents — to all the jobs in C_{OF} and appends the jobs to the end of $C_{RF,m}$. Afterward, the road-to-phase mapping is applied to obtain the inflows.

Each $UpAgent$ executes Algorithm 6 to obtain the planned outflow C_{OF} at the current time cdt, based the previously planned control flow (S^*, C^*_{CF}). The entry road m of the requesting agent is the exit road n of $UpAgent$. In Line 1, (S_{OF}, C_{OF}) is obtained as $(S^*, C^*_{CF}) \cap [cdt, cdt + H_{ext}]$. In Line 3, rr is the road-ratio function, the function $tp(i, m, n)$ is the proportion of traffic turning from the entry road m onto the exit road n during phase i.

For simplicity, Algorithm 5 is described as though an agent can immediately obtain each requesting result. If there are communication delays between agents, Lines 4-5 can be executed later by simply using the segment $C_{OF} \cap [cdt, \infty]$, i.e., the actual horizon extension is just shortened.

A basic property of this protocol is that non-local influences (Witwicki and Durfee 2010) from indirect neighbors can be included if H_{ext} is sufficiently long, since the control flow of direct neighbors contains flow information from their upstream neighbors. A limited H_{ext} is used nonetheless to balance computational cost against the increasing uncertainty of predicted flow information over longer horizons.

The optimistic assumption that is made is that direct and indirect neighbors are trying to follow their schedules. The situation is "perfect" if all upstream neighbors produce output flows precisely according to their schedules (e.g., as when using fixed signal timing plans). Normally, the optimization capability of SchIC makes schedules quite stable, given the clusters in local observation and large clusters (platoons) in non-local observation. However, even if some neighbors change their schedules at their next decision points, those minor changes might still be absorbed by exploiting the temporal flexibility in their control flows.

"Nervousness" Prevention

The first situation of mis-coordination is "nervousness" for a downstream agent due to the uncertainty and disruption associated with the predictions of upstream agents that are using on-line control strategies with finite horizons.

In SchIC, maximum green constraints are not included in obtaining (S^*, C^*_{CF}). This simplification does not present a problem for an isolated intersection, since these constraints can be incorporated by the repair rule when determining and committing to SS. However, when operating within a network, repairs can cause nervousness for a downstream agent due to nontrivial changes between planned and actual outflows from upstream agents.

To avoid a potential disruption, all timing constraints must be incorporated into each planned signal sequence, rather than be repaired after the fact. However, this is difficult to accomplish in SchIC since each cluster is treated as indivisible. Allowing clusters to be dynamically split during the optimization process would require a more sophisticated approach with greater computational complexity.

Instead, a "nervousness" prevention mechanism, shown in Algorithm 7, is added to the coordinated control strategy of each agent. This mechanism iteratively splits clusters to ensure that all maximum green time constraints are satisfied. Based on the current observation o, SchIC is executed (Line 3) to obtain a new solution (S, C_{CF}, PD) for extending (S^*, C^*_{CF}) from the current time cdt (Line 4). Then maximum green time constraints are checked. For each stage $k = 1$ to $|C_{CF}|$, there is a violation if $pd_k > G^{max}_{s_k}$. The violation time point t_{vio} is obtained in Lines 7-8. In Line 9, $(S^*, C^*_{CF}) \cap [t_{vio}, \infty]$ are unscheduled, and IF is updated. In Line 10, t_c, cpi, and cpd in the observation o are updated. The process is repeated until no violation is found.

The number of iterations is equal to the number of violation time points that are found in the schedule. A time violation occurs only when a phase is scheduled to exceed the maximum green time. Given a limited planning horizon,

Algorithm 8 Prevent spillover in the next phase

1: $c_1 = (c \text{ of } mc_1)$; $c_2 = (c \text{ of } mc_2)$; $i_n = next(cpi)$
2: $SOCount = 0$
3: **for** $m \in EntryRoadsServicedinPhase(i_n)$ **do**
4: $\quad SOCount+ = \max(0, |c_2| \cdot rr(c_2, m) - (ac \text{ of } m))$
5: **end for**
6: **if** $SOCount \equiv 0$ **return**
7: $t_{SO} = \min(SOCount/sfr_{i_n}, DelayTime \text{ of } mc_2)$
8: $t_{old} = dep(c_1)$, $t_{new} = \max(0, t_{old} - t_{SO})$
9: $RQCount = |c_1| \cdot (t_{old} - t_{new})/t_{old}$
10: **if** $SOCount \geq RQCount$ **then**
11: $\quad Unschedule(t_{new}, t_{old})$; $Unschedule(dep(c_2), \infty)$
12: $\quad Shift(C^*_{CF} \cap [arr(c_2), dep(c_2)], t_{new} - t_{old})$
13: **end if**

the number of iterations is thus bounded and small.

Spillover Prevention

The second mis-coordination situation is the *spillover* effect (Daganzo 1998). Each road in a traffic network has its arrival capacity (ac), i.e., $ac = L/h$, where L is the road length, and h is the average headway distance between statically queueing vehicles. The spillover due to insufficient capacity on an entry road of a downstream intersection will not only block traffic flow from upstream intersections, but might also lead to dynamic instability (Kumar and Meyn 1995) in a network.

The spillover prevention mechanism is used by a downstream agent to prevent a spillover in the next phase by deciding if the current phase should be terminated earlier than planned. In this mechanism, the downstream agent sacrifices its own interest for the sake of its upstream neighbors.

For the control flow (S^*, C^*_{CF}), all adjacent clusters that are of the same phase are merged into a *macro cluster*, i.e., $mc = (c, PhaseIndex, SlackTime, DelayTime)$, where c is the merged cluster, $PhaseIndex$ is the phase index, $SlackTime$ is the total slack time in mc, and $DelayTime$ gives how long the first cluster in mc will be delayed.

Three conditions are required to trigger Algorithm 8: (1) there is more than one macro cluster; (2) the $PhaseIndex$ of mc_1 and mc_2 are respectively cpi and $next(cpi)$; and (3) $SlackTime = 0$ and $DelayTime > 0$ for mc_2.

If these conditions hold, Algorithm 8 is executed to prevent spillover in the next phase. The basic idea is to obtain an anticipated spillover count $SOCount$ (Lines 2-5) in the next phase, and use the time t_{SO} (Line 7) required for clearing $SOCount$ to estimate the residue queue count $RQCount$ (Lines 8-9) to be sacrificed in the current phase. If $SOCount \geq RQCount$, the actual adjustment to the control flow is performed in Lines 11-12 by shifting clusters in mc_2 ahead to avoid spillover. For simplicity, all unscheduled clusters are discarded, although in principle these clusters might be re-scheduled using SchIC.

Performance Evaluation

To evaluate our approach, we simulate performance on two road networks: (1) a synthetic 5X5 grid network designed to incorporate tightly-coupled intersections with short intervening travel times, high congestion, and a fine time resolution for optimization of $\Delta = 0.5$ seconds, and (2) a real-world traffic network with 2-way traffic, multiple lanes, and multi-directional outflows. The former presents a difficult network optimization challenge in a controlled setting; the latter presents a more complex practical application.

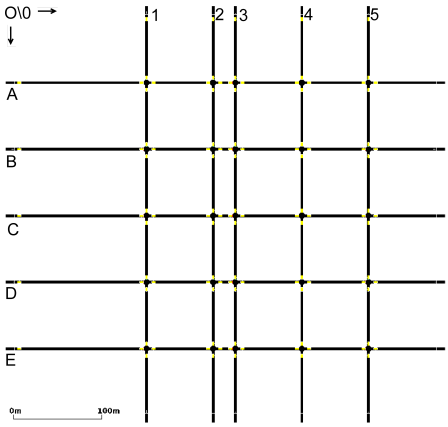

Figure 2: A 5X5 grid network of 25 intersections

All simulation runs were performed using Simulation of Urban Mobility (SUMO) (Behrisch et al. 2011), an open-source micro traffic simulator. For each control strategy tested, we measured performance as the average waiting time of all vehicles, calculated as the mean over 100 runs.

5X5 Grid Network

Simulation Settings The 5X5 grid network is shown in Figure 2. In this network, all roads have length L=75 meters, except for the horizontal roads between **2** and **3**, which have $L_{(2-3)} = 25$ meters, and horizontal entry roads between **0** and **1**, which have $L_{(0-1)} = 150$ meters. The length $L_{(0-1)}$ is used for reducing boundary effects. We will assume that heavier flows are traveling in the horizontal direction.

All roads in the network are one-way. To remove the uncertainty in estimating movement proportions, no turning movements are considered. For background traffic, each minor route generates a traffic flow of $1/20$ of the total traffic. There are two major flows on **C** and **3** (the bottleneck) that generate $3/5$ of the total traffic. The total simulation time is one hour, and for each twenty minute period, the demand ratios between **C** and **3** are 35:25, 40:20, and 45:15.

On each road, the flow speed is 10 meters per second, the saturation flow rate sfr per lane is one vehicle per 2.5 seconds, and the start-up lost time is $slt = 3.5$ seconds. For each intersection, G^{min} and G^{max} are respectively 5 and 55 seconds, and the yellow time is $Y = 5$ seconds.

These settings are fairly realistic. The default value of L is similar to the lengths of east-west streets in Manhattan (about 79 meters). Roads of around 25 meters appear in some real-world urban road networks. The travel times on each road (7.5 seconds for normal roads and 2.5 seconds for horizontal roads between **2** and **3**) are consistent

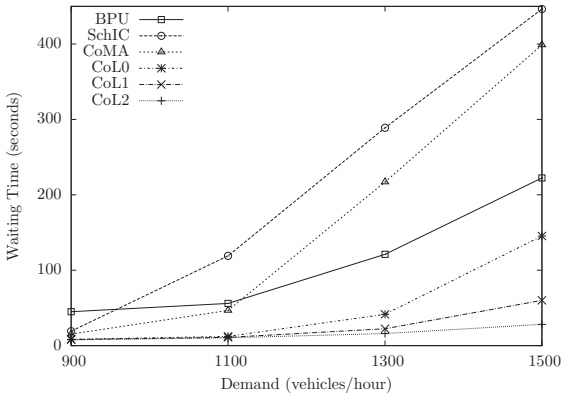

Figure 3: Results of six control strategies

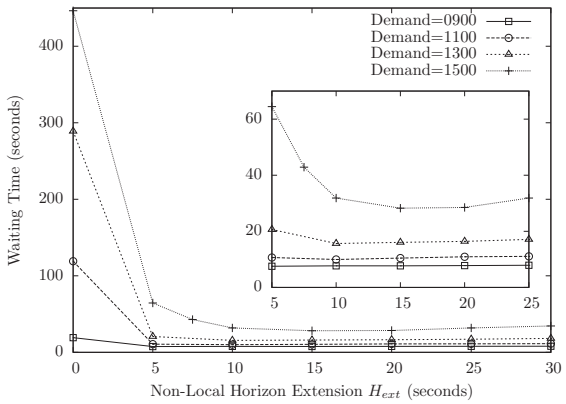

Figure 4: CoL2 with different horizon extensions

with those of real-world urban networks and provide a challenging setting for studying tightly-coupled intersections. Because the minimal time for $SwitchBack$ is 15 seconds ($Y + G^{min} + Y$), non-local impacts from indirect neighbors can be nontrivial.

Five control strategies were configured using the core SchIC intersection control strategy and evaluated: SchIC, CoL0, CoL1, CoL2, and CoMA. SchIC is the isolated application of SchIC with no coordination, which serves as the baseline. CoL0 applies the first coordination mechanism, optimistic non-local observation, to SchIC. CoL1 extends CoL0 by including the second coordination mechanism, nervousness prevention. CoL2 further extends CoL1 by adding the third coordination mechanism, spillover prevention. We also consider CoMA, which is the same as CoL0 except that instead of using real-time data for the non-local horizon extension, we use a moving average of the historical demand on each entry road. The moving average forecast of demand approach is not only adopted by some contemporary online planning approaches (e.g., OPAC), but is also used as a popular mechanism for collaborative planning in supply chain networks (Xu, Dong, and Evers 2001). For SchIC, the parameter for aggregation is set as $th_g = 3$ seconds, and the upper limit for commitment is $th_{ext} = 5$ seconds. By default, the non-local horizon extension H_{ext} is 15 seconds.

A real-time coordination algorithm called *balanced phase utilization* (BPU) (Barlow 2011) is also included for comparison. BPU is representative of another class of adaptive traffic signal control systems, including ACS-Lite. At the end of each cycle, the algorithm adjusts the phase splits in order to balance phase utilization, a measure of phase efficiency. The algorithm also coordinates an intersection by calculating an offset from an adaptively selected neighbor.

Results Figure 3 shows the results of six strategies, i.e., BPU, SchIC, CoMA, CoL0, CoL1, and CoL2, for the demands $\in \{900, 1100, 1300, 1500\}$ vehicles per hour (vph).

Compared to SchIC, CoMA performed better since a basic degree of coordination is introduced by a moving average forecast beyond the prediction horizon.

Due to the strong intersection optimization capability of SchIC, both SchIC and CoMA performed better than BPU when the demand is low. However, as traffic demands increase, non-local impacts cannot be sufficiently addressed by implicit coordination alone. With its explicit coordination through offset calculation, BPU achieved much better performance than SchIC and CoMA for higher demands.

As shown in Figure 3, CoL0 produced lower waiting times than the above three strategies. Compared to SchIC, the advance of CoL0 demonstrates the benefit from the optimistic non-local observation coordination mechanism. CoL0 does not have an explicit offset calculation like BPU; the coordination between neighbors is instead established by looking ahead to upstream outflows. The only difference between CoL0 and CoMA is that the former exploits the importance of demand variability in real-time. Nevertheless, if the communication fails between some neighbors, moving average forecast might be temporarily adopted for those failed links where CoL* and BPU are not functional.

Compared to CoL0, the advance of CoL1 and CoL2 demonstrate the effectiveness of both nervousness and spillover prevention mechanisms. For an upstream agent, nervousness prevention has no local effects on its own performance if there is no spillover from downstream intersections, although it does incur minor computational cost. However, by making the prediction of outflows more reliable, nervousness prevention benefits downstream agents, leading to less mis-coordination. The spillover prevention helps an upstream agent use its green time effectively, though downstream agents may have to sacrifice their own local performance. Our test shows that the sacrifice is worthwhile.

Figure 4 shows the results of CoL2 with different non-local horizon extensions from 0 to 30 seconds at different demands. The sharp increases near $H_{ext} = 0$, especially for high demands, demonstrate the performance advantage of providing non-local information to agents with a myopic view of the environment. At the high demand of 1500 vph, the best result is $H_{ext} = 15$ seconds, which has two nontrivial implications. First, the horizon extension is approximately equal to the travel time across two road segments, which indicates the benefit of incorporating non-local impacts from indirect neighbors. The advantage of optimistic non-local observation is that it can consider those indirect

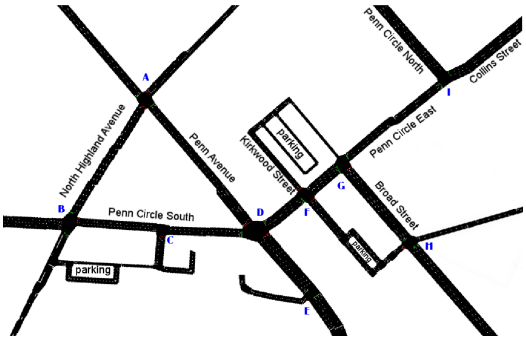

Figure 5: A real-world road network of 9 intersections

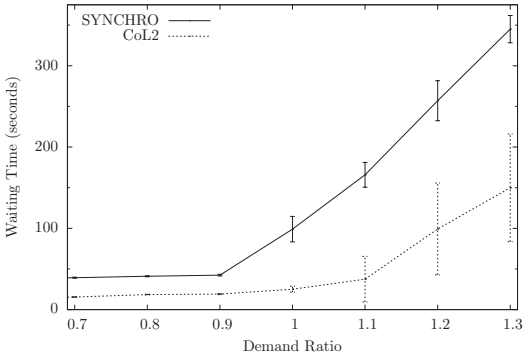

Figure 6: Results on the real-world road network

impacts by using predicted outflows from direct upstream neighbors. Second, the fact that the performance of CoL2 becomes worse when H_{ext} is increased might be due to increased uncertainty in non-local observations. Thus, it might be beneficial to implement a learning strategy to decide H_{ext} for each upstream direction of an agent.

For $H_{ext} = 30$, the number of time steps in the prediction horizon $(H_{ext} + L/speed)/\Delta$ is 75, which presents a substantial look-ahead horizon for existing online planning algorithms. ALLONS-D, for example, becomes intractable at prediction horizons significantly smaller than this.

Of all the instances shown in Figure 4, the instance with H_{ext}=30 seconds and a demand of 1300 vph was the most computationally intensive. For this instance, the total CPU time of CoL2 for each run took between [0.09, 0.41] seconds with an average of 0.22 seconds. Each decision was made, on average, in 1.4E-5 seconds (with the agents taking an average of 15971 decisions per run), which is much shorter than the fine time resolution Δ. All computations were performed using JRE 1.6 on a 3.4GHz AMD Phenom II. Thus, there is still headroom to incorporate and benefit from more sophisticated coordination mechanisms with greater computational requirements on schedule-driven intersections.

Real-World Road Network

Figure 5 shows a 9-intersection road network located in the East Liberty area of Pittsburgh, PA. All roads between intersections are two-way. The number of lanes on each road ranges from 1 to 3, and road lengths range from 40.5 to 174.1 meters. On each road, the flow speed is 13 meters per second. We consider one time-of-day scenario, i.e., an AM peak hour, which has an average demand of 4891 vehicles per hour. For these intersections, the fixed timing plans that are currently used were generated by SYNCHRO, a commercial package for offline traffic signal optimization based on offset calculation, and were provided to us by the city of Pittsburgh. For the head-to-head comparison, only the full SchIC-based strategy (CoL2) is included. CoL2 is applied to each intersection with G^{min} and G^{max} values of 10 and 90 seconds respectively, and all Y values are kept the same as the current operational settings. For model parameters, sfr per lane is one vehicle per 2 seconds, and slt is 2 seconds.

Figure 6 shows the results of CoL2 and the current, fixed coordination strategy optimized by SYNCHRO. Different demand ratios were considered to reflect possible changes of average demands in different days. For all demand ratios, CoL2 achieved significantly better results than the SYNCHRO generated timing plan. Both control strategies produced low waiting times and very low variations when the demands were low. For high demands ratios, the non-linearly increased waiting times indicate that in some runs the network became congested. The variations of CoL2 was large since the adaptive control strategy can sometimes, although not always, avoid congestion.

Conclusions

In this paper, we described a schedule-driven coordination approach for real-time traffic network control. This approach solves the traffic control problem by scheduling aggregate traffic flows in a networked distributed environment. In our model, each intersection is controlled by a distinct agent that uses look-ahead scheduling to operate with a limited prediction of incoming traffic, and this basic control strategy is augmented with three coordination mechanisms to improve overall system performance. In the basic coordination mechanism, each agent uses the planned output flows of vehicles from its upstream neighbors to generate an optimistic observation, which includes look-ahead information from direct and indirect neighbors. In addition to optimistic non-local observation, two other coordination mechanisms help to prevent "nervousness" and dynamic instability in the network.

We compared our method to an isolated control strategy, a coordination strategy that uses a moving average forecast, and a coordination strategy that uses adaptive offset calculation via simulation on a 5X5 grid network designed to present a challenge for decentralized network-wide optimization. The results obtained demonstrated the ability of our method to establish traffic flow coordination with lower average wait times than all competing methods. We also evaluated our approach on a model of a real-world urban road network, showing the ability to outperform the fixed coordination strategy that is currently in use (which was generated offline using SYNCHRO). A pilot test of our approach on this traffic network is planned for early 2012.

There are several aspects of the proposed method that warrant further study. One issue concerns the development of more effective coordination mechanisms. For example, in

the current spillover prevention mechanism, the downstream agent sacrifices its own interest for the sake of an upstream neighbor. However, such a one-sided sacrifice might be futile if the upstream agent changes its schedule at the next decision point to service another phase. It could be interesting to apply pricing mechanisms to dampen the changes made by upstream agents in this context. Another possible improvement might be to introduce negotiation mechanisms (Dudek and Stadtler 2005), although these mechanisms may require additional iterations to reach an equilibrium.

Acknowledgements. This research was supported in part by the Traffic21 Initiative at Carnegie Mellon University, with support from the Hillman Foundation and the Heinz Endowments, and the CMU Robotics Institute.

References

Barlow, G. J. 2011. *Improving Memory for Optimization and Learning in Dynamic Environments*. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Behrisch, M.; Bieker, L.; Erdmann, J.; and Krajzewicz, D. 2011. SUMO - Simulation of Urban MObility: An overview. In *International Conference on Advances in System Simulation*, 63–68.

Blackburn, J.; Kropp, D.; and Millen, R. 1986. A comparison of strategies to dampen nervousness in MRP systems. *Management Science* 32(4):413–429.

Cai, C.; Wong, C.; and Heydecker, B. 2009. Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies* 17(5):456–474.

Cervero, R. 1986. Unlocking suburban gridlock. *Journal of the American Planning Association* 52(4):389–406.

Daganzo, C. F. 1998. Queue spillovers in transportation networks with a route choice. *Transportation Science* 32(1):3–11.

Dudek, G., and Stadtler, H. 2005. Negotiation-based collaborative planning between supply chains partners. *European Journal of Operational Research* 163(3):668–687.

Gartner, N.; Pooran, F.; and Andrews, C. 2002. Optimized policies for adaptive control strategy in real-time traffic adaptive control systems - implementation and field testing. *Transportation Research Record* 1811:148–156.

Gettman, D.; Shelby, S. G.; Head, L.; Bullock, D. M.; and Soyke, N. 2007. Data-driven algorithms for real-time adaptive tuning of offsets in coordinated traffic signal systems. *Transportation Research Record* 2035:1–9.

Heung, T. H.; Ho, T. K.; and Fung, Y. F. 2005. Coordinated road-junction traffic control by dynamic programming. *IEEE Transactions on Intelligent Transportation Systems* 6(3):341 – 350.

Husch, D., and Albeck, J. 2006. *Synchro Studio 7 User Guide*. Trafficware Ltd., Sugar Land, TX.

Jonsson, A., and Rovatsos, M. 2011. Scaling up multiagent planning: A best-response approach. In *International Conference on Automated Planning and Scheduling*, 114–121.

Junges, R., and Bazzan, A. L. C. 2008. Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 599–606.

Kumar, P., and Meyn, S. 1995. Stability of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control* 40(2):251–260.

Lämmer, S., and Helbing, D. 2008. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment* P04019.

Lesser, V.; Decker, K.; and Wagner, T. 2004. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Sytems* 9(1-2):87–143.

Mirchandani, P., and Head, L. 2001. A real-time traffic signal control system: Architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies* 9(6):415–432.

Papageorgiou, M.; Diakaki, C.; Dinopoulou, V.; Kotsialos, A.; and Wang, Y. 2003. Review of road traffic control strategies. *Proceedings of the IEEE* 91(12):2043–2067.

Porche, I., and Lafortune, S. 1999. Adaptive look-ahead optimization of traffic signals. *ITS Journal* 4(3-4):209–254.

Richter, S.; Aberdeen, D.; and Yu, J. 2006. Natural actor-critic for road traffic optimisation. In *Advances in Neural Information Processing Systems*, 1169–1176.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32(1):663–704.

Sen, S., and Head, K. 1997. Controlled optimization of phases at an intersection. *Transportation Science* 31(1):5–17.

Sharma, A.; Bullock, D.; and Bonneson, J. 2007. Input-output and hybrid techniques for real-time prediction of delay and maximum queue length at signalized intersections. *Transportation Research Record* 2035:69–80.

Smith, S. F.; Gallagher, A.; Zimmerman, T.; Barbulescu, L.; and Rubinstein, Z. 2007. Distributed management of flexible times schedules. In *International Conference on Autonomous Agents and Multiagent Systems*, 484–491.

Witwicki, S. J., and Durfee, E. H. 2010. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *International Conference on Automated Planning and Scheduling*, 29–36.

Wu, F.; Zilberstein, S.; and Chen, X. 2009. Multi-agent online planning with communication. In *International Conference on Automated Planning and Scheduling*, 321–328.

Xie, X.-F.; Smith, S. F.; Lu, L.; and Barlow, G. J. 2011. Schedule-driven intersection control. Technical Report CMU-RI-TR-11-34, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Xu, K.; Dong, Y.; and Evers, P. T. 2001. Towards better coordination of the supply chain. *Transportation Research Part E: Logistics and Transportation Review* 37(1):35–54.